Mobila appliaktioner och trådlösa nät, HI1033, HT 2012

Today:

- Challengers with mobile services
- Platforms
- Android

What is a Mobile Service?

What is a Mobile Service?

Mobile devices

Pico	Pocket	Palm	Pad	Lap	Desk
	Mobile	*PDA*	*E-reader*	Laptop	PC
	Smartphone	Net-book			
	Tablet				

Smartphone vs Feature phone

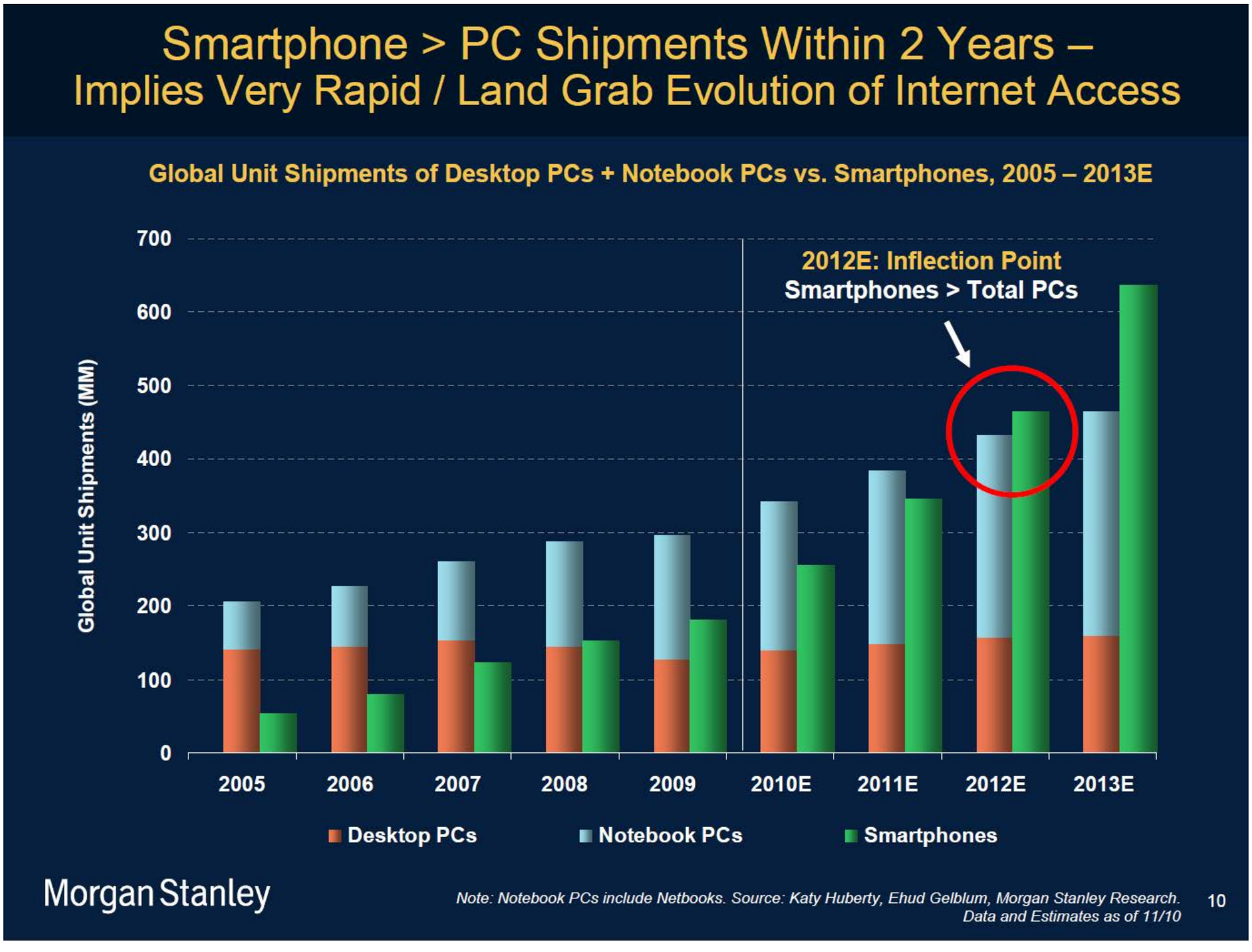
- Smartphone - “A handheld computer integrated within a mobile telephone”

- Smartphones run complete operating system software providing a platform for application developers

- Common features (italics: usually *not* on a feature phone)
  - Play media
  - Connect to internet
  - *Touch screen*
  - *Hard/Soft keyboard*
  - Run third party software (e.g. J2ME or “apps”)
  - *Run third party software written in a native language*
  - *Additional devices like WiFi, GPS, accelerometer, …*
  - *Access to hardware*

Market

Smartphone > PC Shipments Within 2 Years – Implies Very Rapid / Land Grab Evolution of Internet Access



Morgan Stanley

Note: Notebook PCs include Netbooks. Source: Katy Huberty, Ehud Gelblum, Morgan Stanley Research. Data and Estimates as of 11/10

Market

Nu säljs det fler smarttelefoner än datorer

Smarttelefonerna gick i fjol om persondatorerna, sett till den totala försäljningen världen över.

MARTIN WALLSTRÖM
martin.wallstrom@idg.se

Under 2011 såldes det fler smarttelefoner än datorer, för första gången, enligt siffror från analysföretaget Canalys.

Totalt såldes 488 miljoner smarttelefoner under hela året. Antalet persondatorer, här ingår pekplattor, bärbara och stationära datorer, var 415 miljoner.

Bärbara datorer stod för mer än hälften av datorförsäljningen, medan plattor utgjorde 15 procent.

Antalet sålda smarttelefoner ökade med 63 procent under året. Den globala pc-marknaden ökade 15 procent och det är pekplattor som står för merparten av ökningen.

Canalys spår att försäljningen av smarttelefoner ökar långsammare i år. Den totala försäljningen väntas ändå överstiga en halv miljard. Framförallt billigare smarttelefoner väntas sälja mer än tidigare.

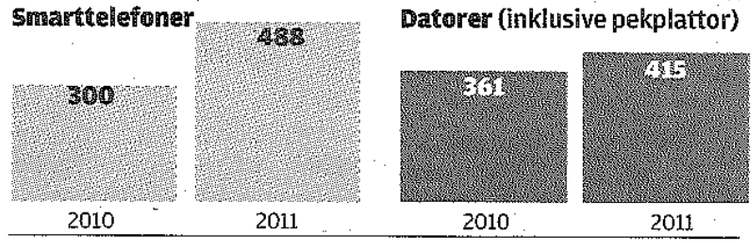
Apple är den största leverantören av smarttelefoner, följt av Samsung, Nokia och RIM.

FAKTA

Datorerna tappar mark

Antalet sålda smarttelefoner och datorer per år.
(miljoner enheter)

KÄLLA: CANALYTICS



Computer Sweden, februari 2012

Some platforms

- iPhone, iPad (derived from Mac OS X, Unix-like). API: Objective C.

- Symbian OS (derived from EPOC). API: C++.
  Open source, today maintained by Nokia.

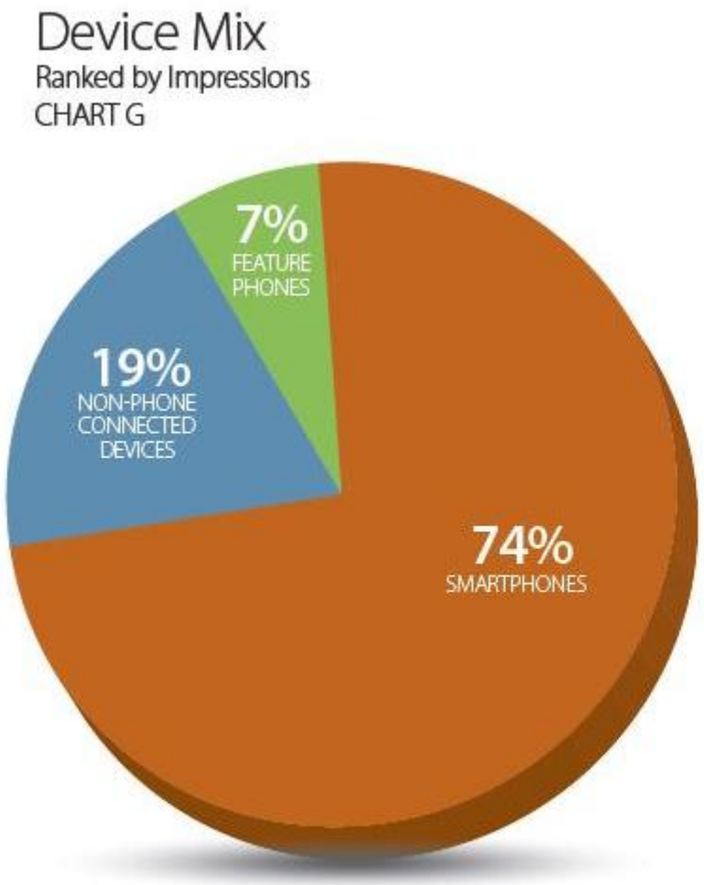
- Android. Linux kernel + Dalvik Virtual Machine running applications. API in Java dialect.
  Open source, maintained by Open Handset Alliance

- Windows Phone. Operating system with 3rd party and Microsoft services.

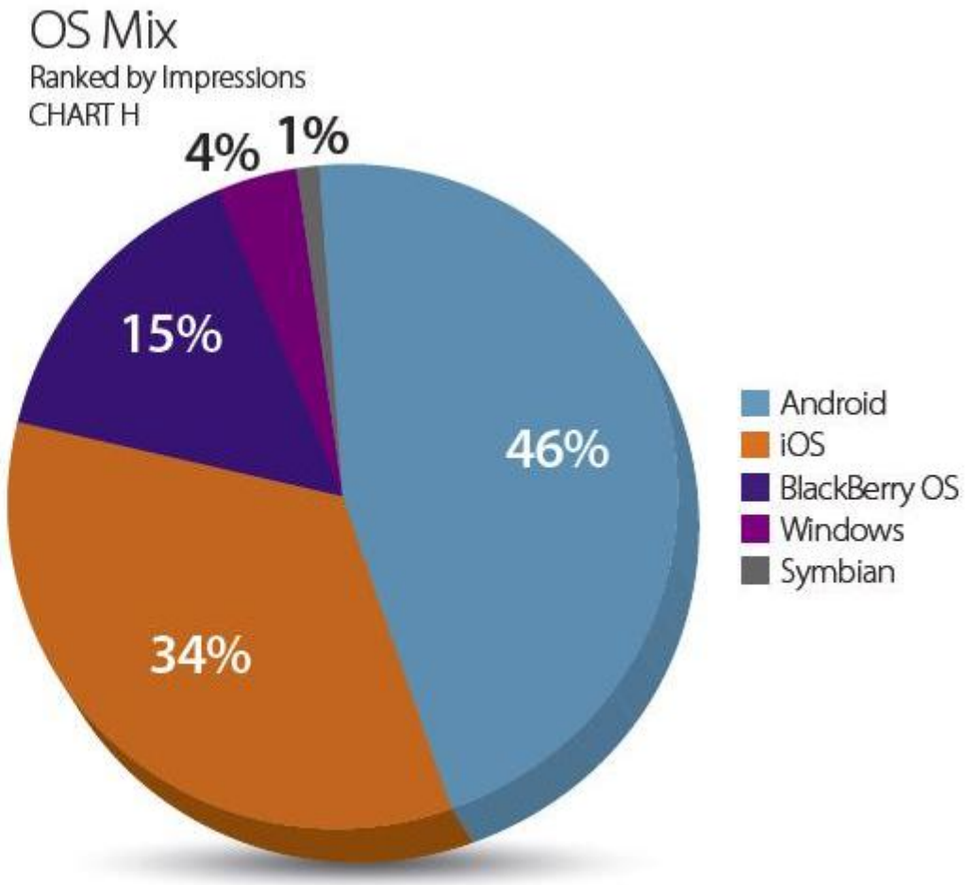
- Java Micro Edition: Cross platform; runs on a virtual machine on top of other OS. Designed for embedded systems. Down-scaled Java API.

Some Smartphone platforms

Q2 2012. Källa: Millennial Media



Source: Millennial Media, Q2 2012.
Smartphone data does not include what could be considered Smartphones running proprietary Operating Systems, e.g. Samsung Instinct, LG Vu



Source: Millennial Media, Q2 2012.

Market; App Stores

- Revolution in distribution of mobile applications.

- Applications available for download “over the air” (June 2011)
  - App Store: 400 000 (from approximately 30 000 developers)
  - Google Play (tidigare Android Market): 400 000
  - Windows Phone Marketplace: > 20 000
  - BlackBerry AppWorld: > 30 000

- App Store 2009:
  Every app store user spends an average of €4.37 every month.
  There is over 58 million app store users.

Typical Smartphone specs

	iPhone 4	Motorola Droid X	Typical PC
Mass storage	16-32 GB	24 GB (8 GB internal, 16 GB microSD)	1 TB
RAM	512 MB	512 MB	4 GB
Processor	Apple A4, 1GHz	Texas Instruments ARM, 1 GHz	3-3.5 GHz*
Battery Stand by/Talk	*300 hours/420 minutes*	*220 hours/480 minutes*	*-*

Expect this when developing software for limited devices such as smartphones

- Limited memory capacity and processor speed
- Network: High latency, low speeds. Might be associated with a cost(!)
- Small screens, of different sizes
- Application might get interrupted at any time(!)
- Hardware-imposed design considerations
- *Design with this in mind:*
  Be efficient and be responsive

What’s consuming memory, processor resources and battery?

- Memory
  - Unnecessary allocation of objects
  - Inefficient data structures
  - Size of application code(!)
  - Multiple processes

- Processor recources
  - Inefficient algorithms
  - Garbage Collection(!)
  - Multiple processes and threads
  - Rendering of GUI
  - Unnessecary polling

- Battery
  - Processor working
  - Network communication, especially when using WiFi

Mobile Internet Services

Telecom

- GSM, GPRS, EDGE, 3G and 4G
- Network and Services is often connected

Datacom

- Local IEEE 802.11 networks (WiFi)
- Network and Services is separated

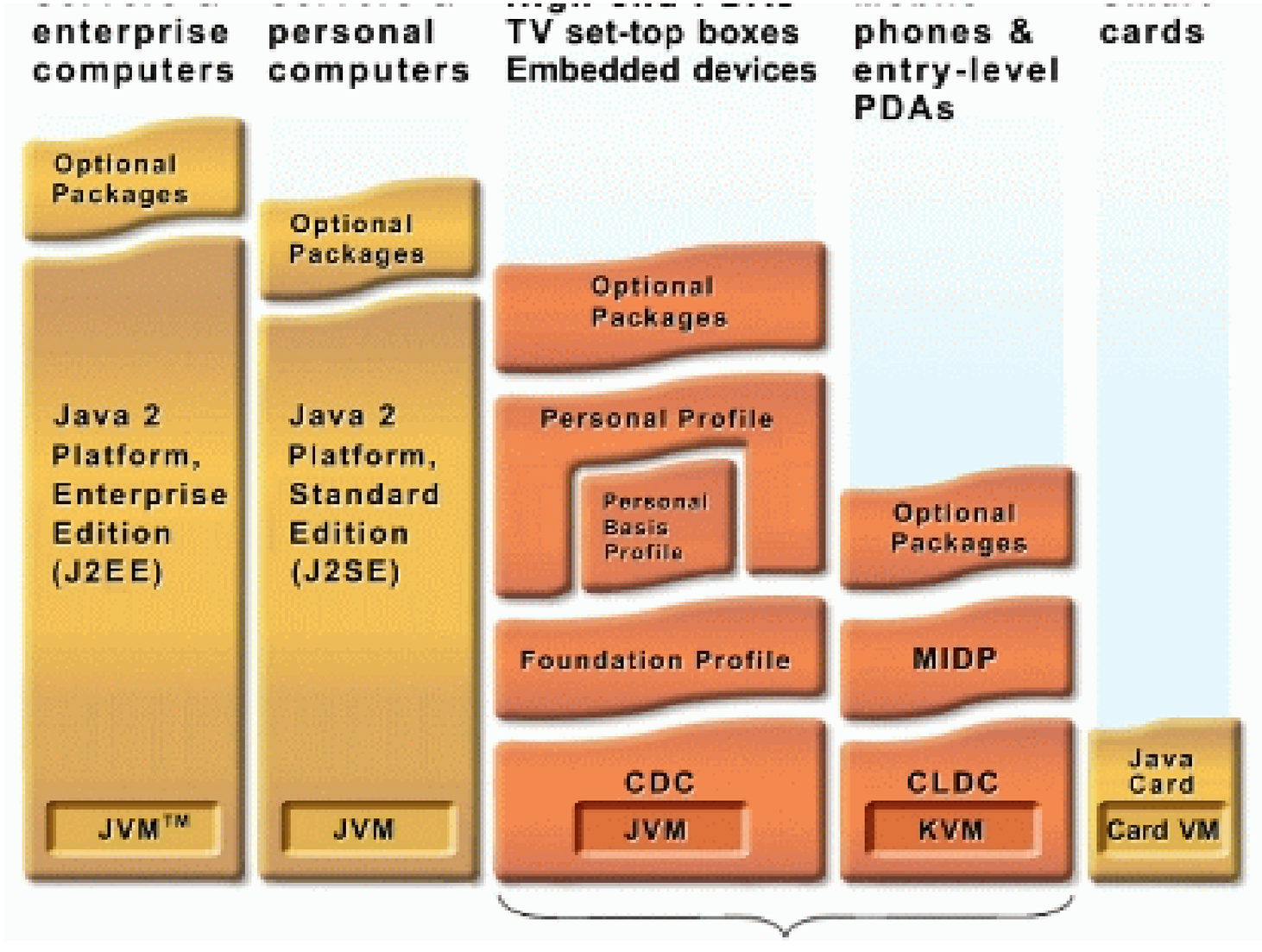
Challenges with mobile data

- Low bandwidth, Frequency vs. Bandwidth
  - GSM, GPRS, EDGE, 3G/4G, WLAN, LAN
  - Wireless connection using different networks
- Datacom vs. Telecom - Best effort vs. Quality of Service
  - Cost and distance
  - Push vs. Pull
- Question regarding benefit, design and standards

Java Micro Edition

- In the middle of the 90s OAK was developed (Java predecessor)
  1999 Palm included KVM (Kilobyte Virtual Machine)

- Supposed to work on:
  - “high-end” PDA and cellular phones
  - set-top boxes, TV, embedded
  - smart cards



Java Micro Edition, Limitations

- Limitations that have had great impact on the development of Java ME:
  - Capacity; CPU and Memory
    - 16 MHz to 1G Hz
    - 128 kb to 128 Mb RAM
  - Power
  - Connection
  - User Interface and physical form

CLDC and CDC

- Two different Java ME configurations:
    - CLDC (Connected Limited Device Configuration) Focus on the most limited devices
    - CDC (Connected Device Configuration) Devices that almost handle a complete Java environment
- Why:
    - One common ground for similar devices
    - Keep “core” API’s between different devices
    - Define requirements on virtual machines

Java Micro Edition

- There are billions of Java ME enabled mobile phones and PDAs

- Java ME might become an old technology, as it is not used on any of today's newest mobile platforms;
  e.g. iPhone, Android, Windows Phone 7, BlackBerry's new QNX

- http://www.oracle.com/technetwork/java/javame/overview/index.html

At last…

- Android is: A mobile device platform including an OS based on the Linux kernel, middleware and key applications

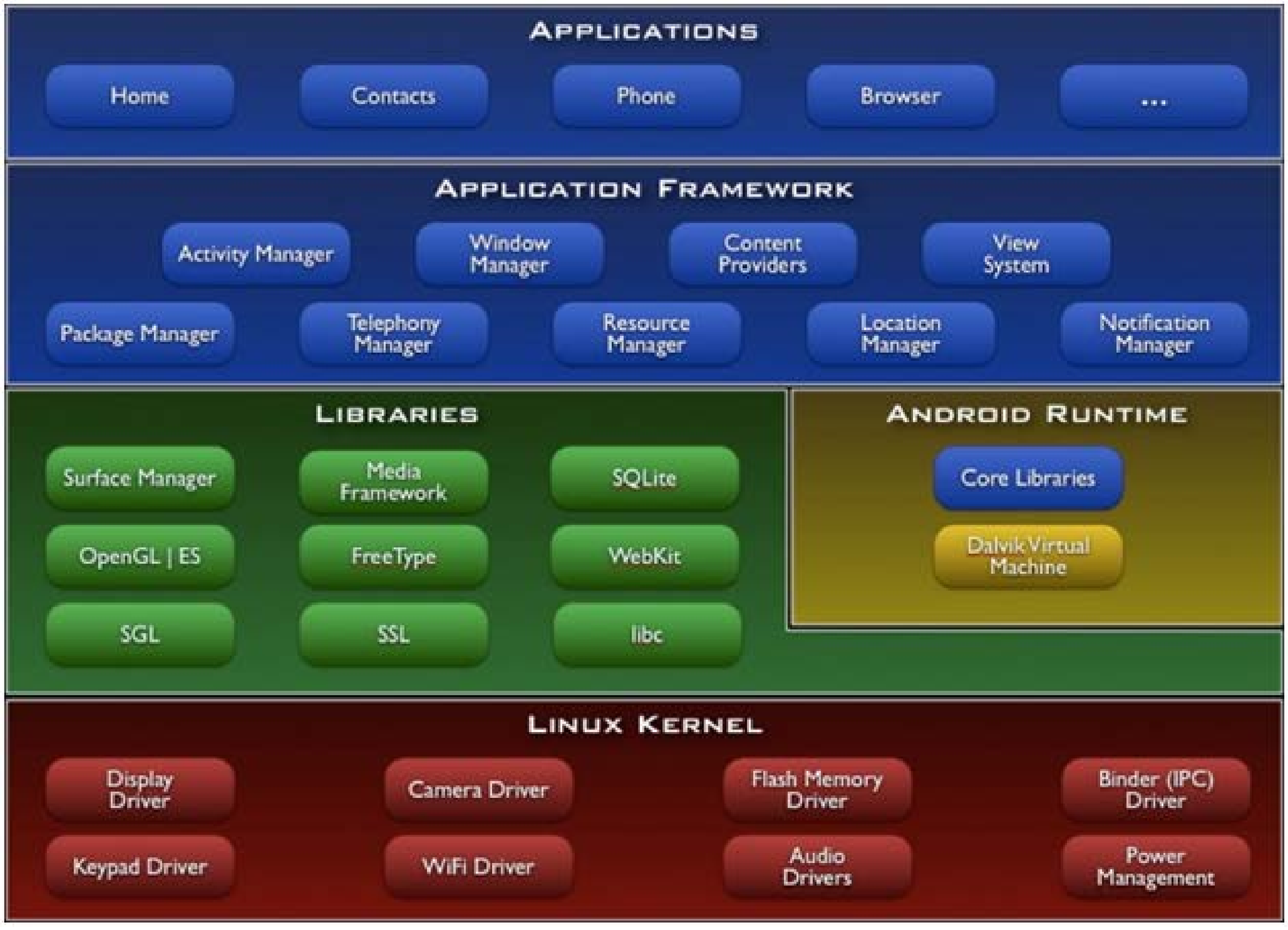
- Designed to support many different hardware devices

- Applications run on the Dalvik Virtual Machine

- An extensive API, including most Java SE classes, for 3rd party application development

- Available under a free software / open source license (no license cost) Standard maintained by Open Handset Alliance, a consortium including Texas Instruments, Google, HTC, Intel, Motorola, SonyEricsson, Samsung, ...

The Android Software Stack



The Dalvik VM

- Every Android application runs in its own process, with its own instance of the Dalvik virtual machine.

- And, yes, Dalvik has been written so that a device can run multiple VMs efficiently.

- The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint

- JIT, Just-In-Time compilation enhance performance (since Android 2.2)

- Android starts the process when any of the application's code needs to be executed.
  The process is shut down when it's no longer needed *and system resources are required by other applications*

Android applications

- Android applications don't have a single entry point (no main method)
  Instead: Consists of essential *components* that the system can instantiate and run as needed

- **Activities** presents a visual user interface (holding View components)

- **Services** doesn't have a visual user interface, but rather runs in the background

- **Broadcast receivers** receive and react to broadcast announcements, e.g. battery is low

- **Content providers** makes a specific set of the application's data available to other applications

Android applications, Activities

- When the first of an application's components needs to be run, Android starts a Linux process for it with a single thread of execution.

- By default, all components of the application run in that process and thread.

An activity has essentially three states:

- *active* or *running* when it is in the foreground

- *paused* if it has lost focus but is still visible to the user

- *stopped* if it is completely obscured by another activity

Android applications, Activities

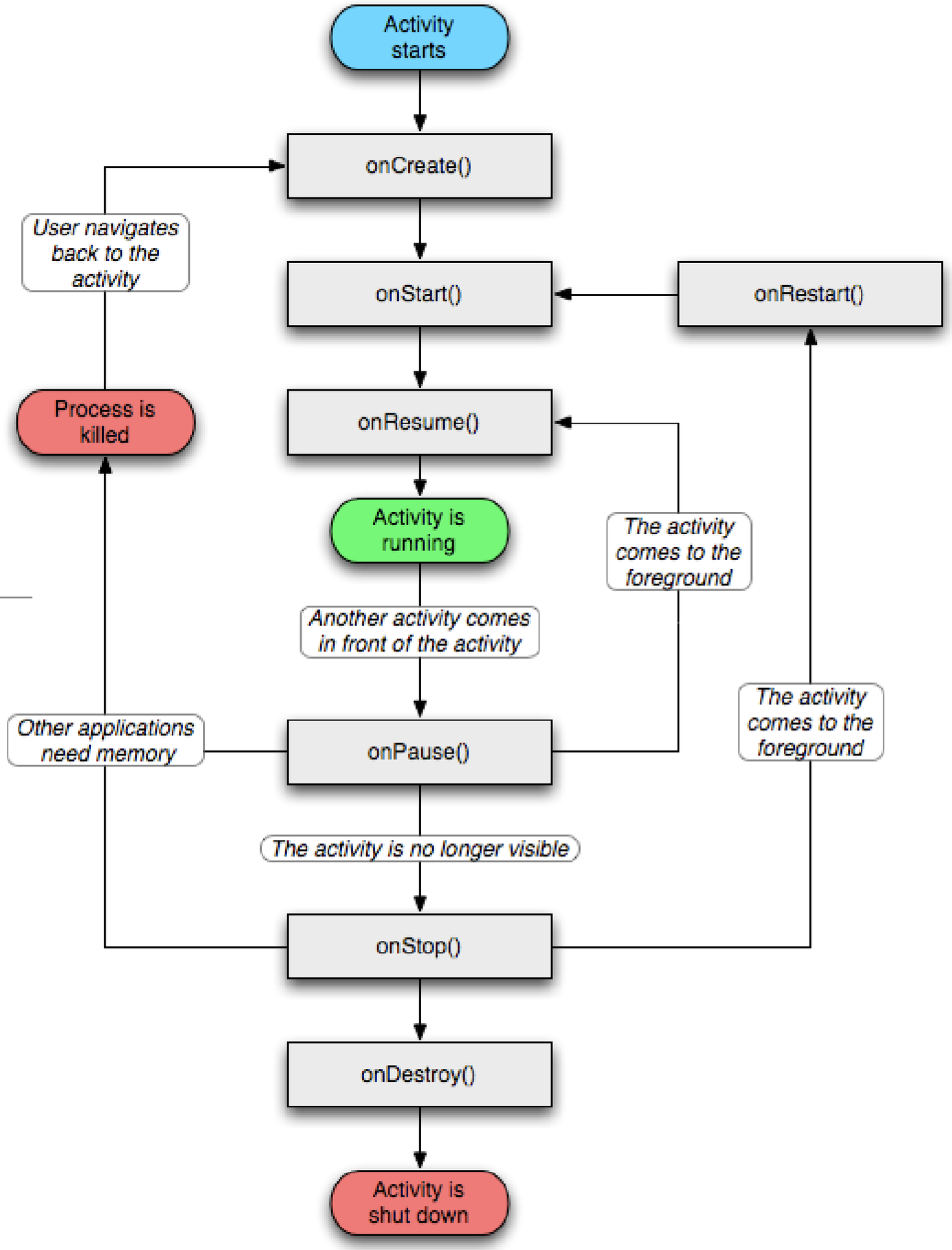
- A paused or stopped activity retains all state and member information, however…

- *…the system can drop it from memory when memory is needed elsewhere*

- As an activity transitions from state to state, it is notified of the change by calls to the following protected methods:

- `void onCreate(Bundle savedInstanceState)`
  `void onStart()`
  `void onRestart()`
  `void onResume()`
  `void onPause()`
  `void onStop()`
  `void onDestroy()`

Activity lifecycle

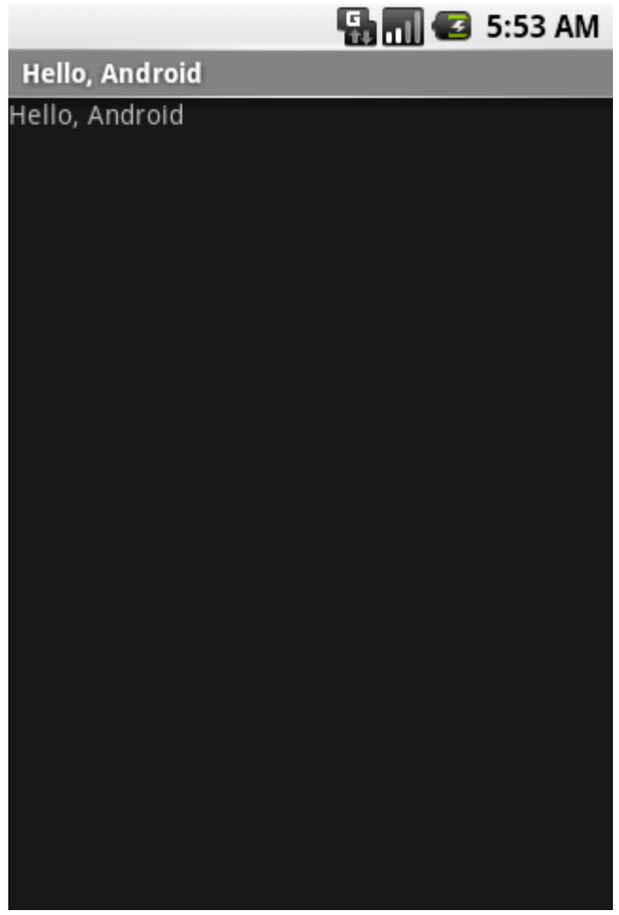


Android applications, Activities

```
package se.kth.hello;

import android.app.Activity;
import . . .;

public class HelloAndroid extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText("Hello, Android");
        setContentView(tv);
    }
}
```



Alternative: layout defined in layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, Android" />

</LinearLayout>
```

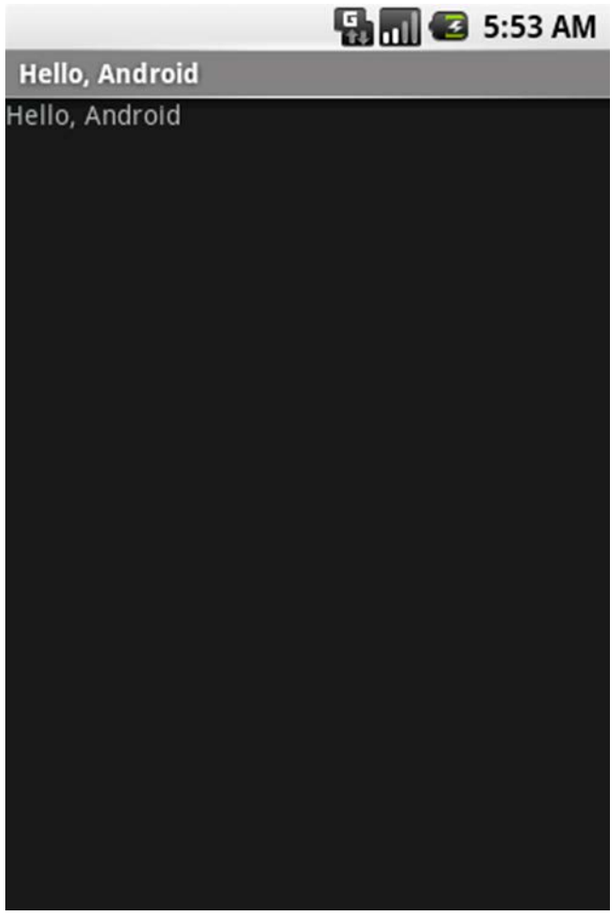
Alternative (cont): “load” layout in actvivity

```
package se.kth.hello;

import android.app.Activity;
import . . .;

public class HelloAndroid extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.main);
        TextView tv = (TextView) this.findViewById(R.id.textView);
        tv.setText("Hello, Android");
    }
}
```



Android from the developer perspective

- High level Java APIs for accessing hardware such as camera, GPS, accelerometer – same interface for different devices

- Native and 3rd party applications are treated equal. You may
  - replace native applications
  - access the same underlying data and hardware
  - use components of native applications

- Reuse of application components (Activities) in other applications possible

- Support for background services

- WebKit, persistent storage using SQLite, OpenGL, …

Android from the perspective of the developer

APIs including

- WiFi hardware access. GSM and 3G for telephony or data transfer
- GPS
- Bluetooth
- HTML 5 WebKit-based browser
- Hardware accelerated graphics (if possible) including OpenGL
- And more…

Some ”Designing For Performance” guide lines

- Memory management
  - Avoid creating unnessecary objects
  - When concatenating text in a loop – use a StringBuffer instead of Strings

- Minimize (virtual) method calls
  - Avoid internal use of getters and setters
  - Declare methods that don’t access member fields as ”static”

- Use the ”for-each” loop except for arrays and ArrayLists

- Know and use the API-libraries – they are probably more efficient than your custom code (e.g. animations)

- Use events +callbacks methods instead of polling for data

Android – where to go from here?

- This is where you find it all:
  http://developer.android.com/index.html

- More on developing for performance:
  Meier, pp 30-38
  http://developer.android.com/guide/practices/design/performance.html